LLNL-SM-484703

Lawrence Livermore National Laboratory

Visualization with VisIt Class Exercises Part II

B. J. Whitlock

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Introduction

This document contains the exercises for the *Visualization with VisIt* class. The exercises are broken down into sets that can be completed as the class progresses and you become more familiar with using VisIt. The VisIt class allows time for the first few exercises in each exercise group, making the remaining exercises optional.

Conventions

This document will usually give the exact steps needed to complete the exercises but exercises will get more difficult and sometimes you will only be given the name of a database and an image of a plot, leaving you to figure out how to recreate the image using your acquired knowledge of VisIt's features. Some exercises within an exercise group require the successful completion of the previous exercise.

Text formatting conventions

- Names of controls or windows in VisIt's user interface will appear in **bold** print but words like: window, button, list, and menu will not usually be printed in bold print.
- When you are asked to type text, the text will be *italicized.*
- Filenames will be *italicized.*

Data directory convention

- This class uses many data files in a directory called *VisItClassData*.
- The *VisItClassData* directory may be installed in various locations such as C:\ or on the Desktop. It depends on how the class materials have been installed for your training session.
- The exercises herein will refer to the actual installed location of the *VisItClassData* directory as VISITCLASSDATA and you will need to substitute the appropriate directory when you need to navigate to the data directory.
- The instructor will tell you where data files were installed.
- The instructions may refer to subdirectories under the VISITCLASSDATA directory and will do so by appending a subdirectory name to VISITCLASSDATA (e.g. VISITCLASSDATA\shapefile).

Organization

The organization of this exercise guide roughly follows the topics discussed in the Visualization with VisIt class, though sometimes new concepts may be introduced before they have been discussed in the class for the sake of providing more interesting exercises.

Tips before you begin

- It might be helpful if you close the compute engine every once in a while after making it do a lot of computations to free up memory and resources. Closing the compute engine is especially important if you notice that your Windows desktop computer becomes sluggish.

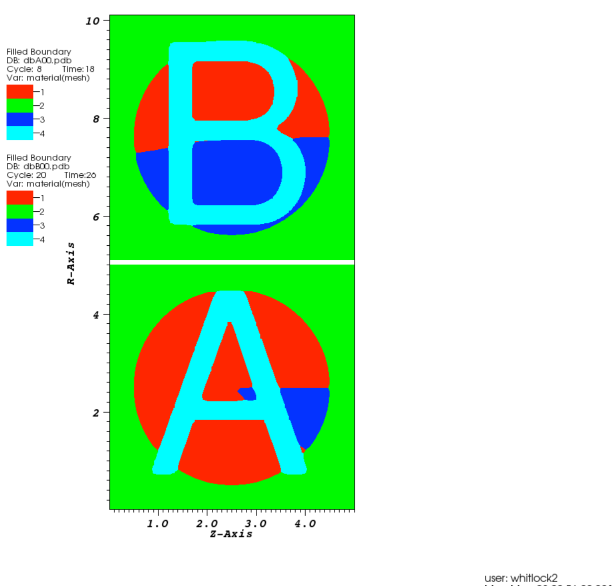
Exercise group 11: Data Comparison

The exercises in this section will give you experience opening multiple databases at the same time and comparing them.

Exercise 11a) Using Multiple Time Sliders

This exercise teaches you how to open multiple time-varying files in VisIt and use multiple time sliders to control plots independently.

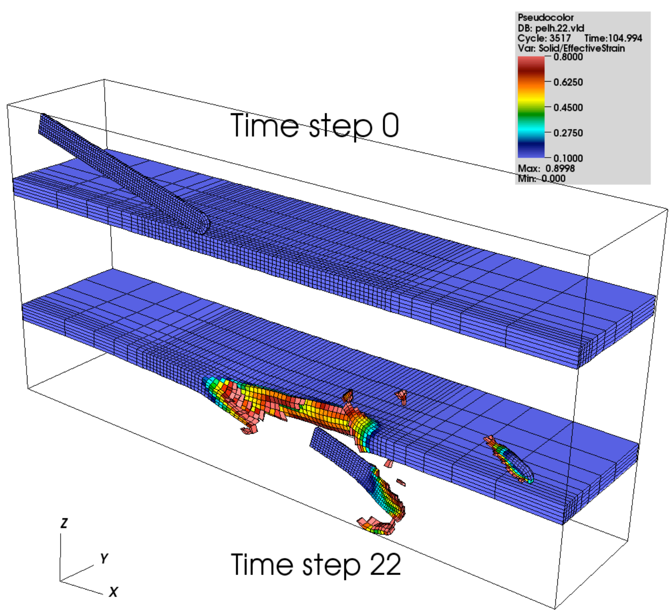
1. Open *dbA00.pdb*.
2. Create a FilledBoundary plot of *material(mesh)* and click **Draw**.
3. Open *dbB00.pdb*.
4. Create a FilledBoundary plot of *material(mesh)* and click **Draw**.
5. Note how the **Time** controls in the **Main** window now feature an **Active time slider** menu since there is more than one time slider.
6. Change the time slider for *dbB00.pdb* and see how only the plot from that database updates.
7. Change the active time slider to *dbA00.pdb*.
8. Change the time slider for *dbA00.pdb* and see how only the plot from that database changes.
9. Delete all plots.
10. Close *dbA00.pdb* and *dbB00.pdb*.



Exercise 11b) Disconnecting Plots From Time Slider

This exercise demonstrates how to create plots from a time-varying database and disconnect them from the time slider so they do not update when you change time states.

1. Open *VISITCLASSDATA\velodyne\pelh.*.vld database*.
2. Add a Pseudocolor plot of *Solid/EffectiveStrain*.
3. Set the plot's **Minimum** value to *0.1* and its **Maximum** value to *0.8.*
4. Set the plot's **Color table** to *hot_desaturated*.
5. Create a Mesh plot of *Solid*.
6. Select the Pseudocolor plot and right click on it to activate the context menu. Next, select **Clone** to create a copy of the plot.
7. Select the Mesh plot and clone it using the **Clone** option in the context menu.
8. Turn off the **Apply operators** check box under the **Plot list** in the **Main** window.
9. Select the last 2 plots in the **Plot list** and add a Transform operator to them.
10. Open the **Transform operator attributes** window and translate the plots *-5* units in the **Z** dimension.
11. Select the first plot in the plot list and right click to open the context menu. Click on the Disconnect from time slider option to make the plot ignore any further changes to the time sliders.
12. Select the second plot in the plot list and disconnect it from the time slider too.
13. Move the time slider to a later time state such as *0022* so you can see one set of plots update while the first of plots continue to show data from time state 0.
14. Delete all plots.

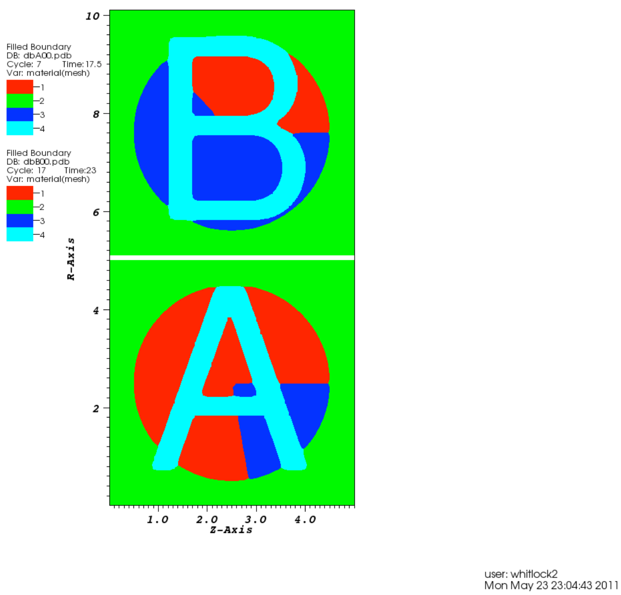


Exercise 11c) Creating a Database Correlation

Now that you have used multiple time sliders to control how plots from different databases behave with respect to time, you can create database correlations to relate two or more databases using a single time slider.

1. Open *dbA00.pdb.*

2. Create a FilledBoundary plot of *material(mesh)* and click **Draw**.
3. Open *dbB00.pdb*.
4. Create a FilledBoundary plot of *material(mesh)* and click **Draw**.
5. Open the **Database Correlation** window from the **Controls** menu.
6. Click the **New…** button to create a new database correlation using the **Create database correlation** window.
7. Select both *dbA00.pdb* and *dbB00.pdb* from the list and click the right arrow button to move them over into the list of **Correlated sources**.
8. Select **Time** for the **Correlation method** so the time values from the databases are used to line up their time steps within the new database correlation.
9. Click **Create database correlation**. Note how the **Active time slider** in the **Time** controls is now set to the *Correlation01* time slider. This is a new time slider that corresponds to the database correlation and is used to control both *dbA00.pdb* and *dbB00.pdb* plots.
10. Change the time state using the time slider in the **Time** controls and watch both plots update.
11. Delete all plots.

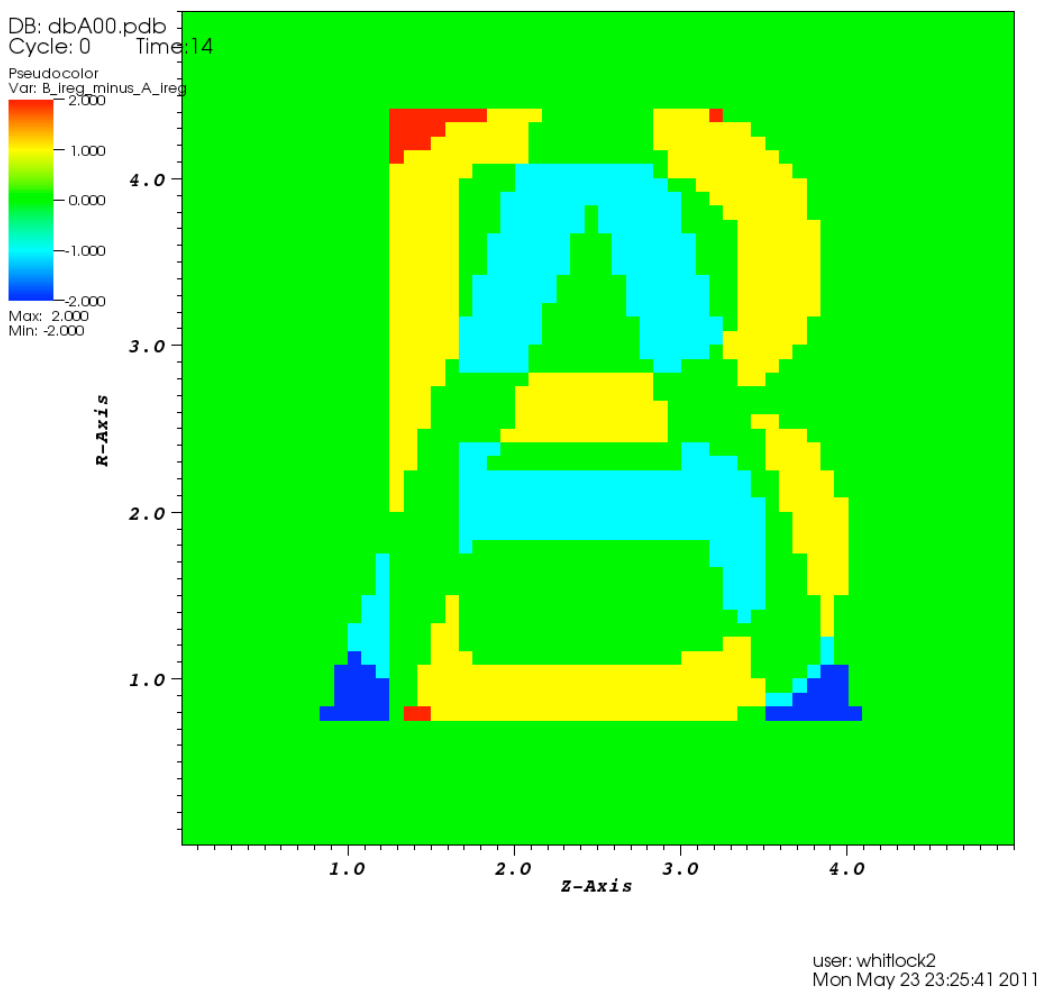


Exercise 11d) Data Level Comparison Wizard

Using CMFE expressions to compare databases can be difficult since they are complex to set up. To simplify this process, VisIt provides a Data Level Comparison Wizard that guides you through a series of questions. The wizard then generates the CMFE expressions needed to accomplish the type of comparison you want to perform.

1. Open *dbA00.pdb*.
2. Open *dbB00.pdb*
3. Open the **Data Level Comparison** wizard from the **Main** window's **Controls** menu.

4. Choose **Between meshes on two separate databases** and advance to the next wizard page.
5. Click *dbA00.pdb* for the **Target Database**.
6. Select mesh for the **Target Mesh**.
7. Select *dbB00.pdb* for the **Donor Database**.
8. Select *mesh/ireg* for the **Donor Field**. This will map dbB00.pdb's mesh/ireg field onto dbA00.pdb's mesh.
9. Advance to the next wizard page.
10. We know these 2 datasets have the same connectivity but exist in different physical spaces so we want **Connectivity-based CMFE**.
11. Advance to the next wizard page.
12. Set the **Name of Expression** to *B_ireg*.
13. Finish the wizard.
14. Open the **Expression** window.
15. Create a new expression *B_ireg_minus_A_ireg = B_ireg - <mesh/ireg>*
16. Create a Pseudocolor plot of *B_ireg_minus_A_ireg* and **Draw**. You will see imprints of both "A" and "B" in the resulting plot.



Exercise group 12: Python Scripting

VisIt provides multiple client interfaces that are capable of controlling the viewer program, which shows plot geometry generated by VisIt's compute server. One such client interface is VisIt's Python interface. The Python interface is a simple program that contains an embedded Python interpreter that has been extended to supply functions to drive VisIt's viewer. VisIt's GUI and Python interface can be connected to the viewer at the same time and both can drive the viewer, causing updated state to appear in the other client. VisIt's GUI even provides a command window that can send Python commands that you type to VisIt's Python interface where they can be used to drive the viewer.

Exercise 12a) Executing Scripts

VisIt's Python interface (CLI) is a Python interpreter that imports the VisIt Python module when it starts up. This allows what is an ordinary Python interpreter to control VisIt via scripting by calling functions that give VisIt instructions to open databases, create plots, etc. You can enter Python commands into VisIt's **Command** window in order to execute Python scripts.

1. Open *noise.silo*.
2. Create a Pseudocolor plot of *grad_magnitude*.
3. Add a Slice operator and turn off its **Project to 2D** check box.
4. Click the **Draw** button.
5. Now for a small demonstration of why scripting is so powerful. Type the following code into the **Command** window:

```
s = SliceAttributes()
s.originType = s.Percent
for i in xrange(0,100,5):
     s.originPercent = i
     SetOperatorOptions(s)
```

6. Click the **Execute** button in the **Command** window to make VisIt interpret the script.
7. The above code should make a slice plane move through the data in 3D.
8. To make the slice plane move through each dimension, type the following code into the **Command** window:

```
normals = ((1,0,0), (0,1,0), (0,0,1))
for n in normals:
     s.normal = n
     for i in xrange(0,100,2):
         s.originPercent = i
         SetOperatorOptions(s)
```

9. Click the **Execute** button in the **Command** window to make VisIt interpret the script.

Exercise 12b) Command Recording and Macros

All of the operations that are possible in VisIt's GUI are translated into commands that are sent to VisIt's viewer process. Most of these commands have an analogue in VisIt's Python interface. VisIt's Python interface is capable of translating the commands it receives from various VisIt clients into Python commands that can be logged or recorded. This exercise teaches you how use VisIt's Python interface to record actions taken in the GUI into Python scripts.

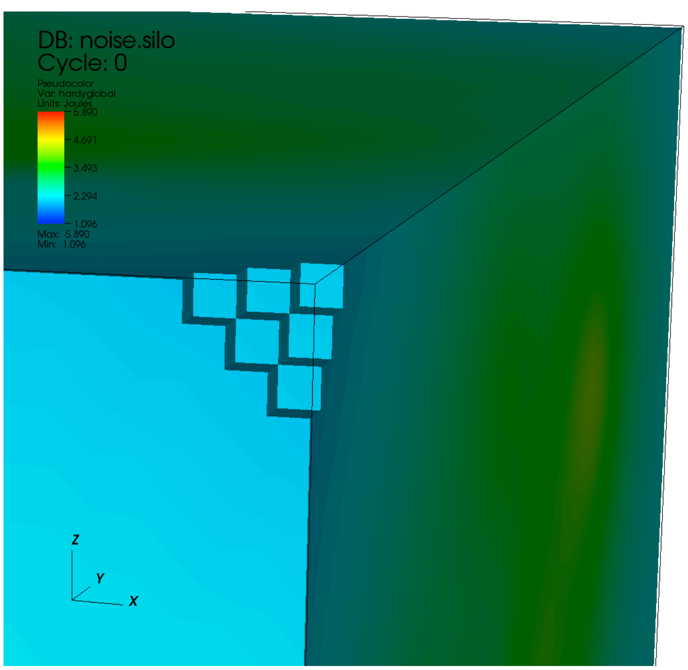
1. Open the **Command** window.
2. Click on the **Record** button.
3. Open *noise.silo*.
4. Add a Pseudocolor plot of *grad_magnitude*.
5. Add a ThreeSlice operator.
6. Click **Draw**.
7. Reset the view.
8. Go back to the **Command** window and click the **Stop** button to stop command recording. The Python commands for the actions described in this exercise will be printed to the active tab in the **Command** window. You can use this technique to figure out the Python code needed to execute operations in complex GUI procedures.
9. Click the **Make macro** button. When you are prompted for the name of a macro, enter *My testing macro*. Be sure to avoid adding any punctuation character since they may cause invalid Python code to be generated.
10. Once you create the macro, the **Commands** window will switch to its macro tab where it has defined a function called *update_macro_My_testing_macro* that contains the body of code that you had recorded. Also note the call to *RegisterMacro* which tells VisIt to create a button in the **Macro** window that is associated with the new macro function.
11. Click the **Update macros** button.
12. Delete all plots.
13. Open the **Macro** window in the **Main** window's **Controls** menu.
14. Click the **My testing macro** button to execute your macro and set up the plot with the ThreeSlice operator.

Exercise 12c) Python Callbacks

This exercise teaches you how to write Python callback functions that get called in response to actions that you take in VisIt. This example combines pick, the map expression, and the Threshold operator to implement a feature that removes each cell that you click on using zone pick. The VisIt function that registers callback functions is called *RegisterCallback*.

1. Open *noise.silo*.
2. Create a Pseudocolor plot of *hardglobal* and click **Draw**.
3. Zoom in on one of the corners of the dataset.
4. Open the **Command** window and select an empty tab.
5. Locate the *VISITCLASSDATA\rmpickcells.py* Python script and open it in a text editor. Select all of the Python script and copy and paste it into VisIt's **Command** window.
6. Click the **Execute** button in the **Command** window to execute the script. This will cause a Python callback function to be registered with VisIt and the function will be called each time the Python interface receives a PickAttributes object from the viewer.
7. Go back to the visualization window and enter **Zone pick** mode.
8. Click on the cells next to the corner of the dataset. Each cell that you click on will be removed from the visualization.
9. Study the comments in the *rmpickcells.py* script so you can get a sense of what each part of the script does. For more information about Python scripting with VisIt, see the *VisIt Python Interface Manual* or *http://www.visitusers.org*.

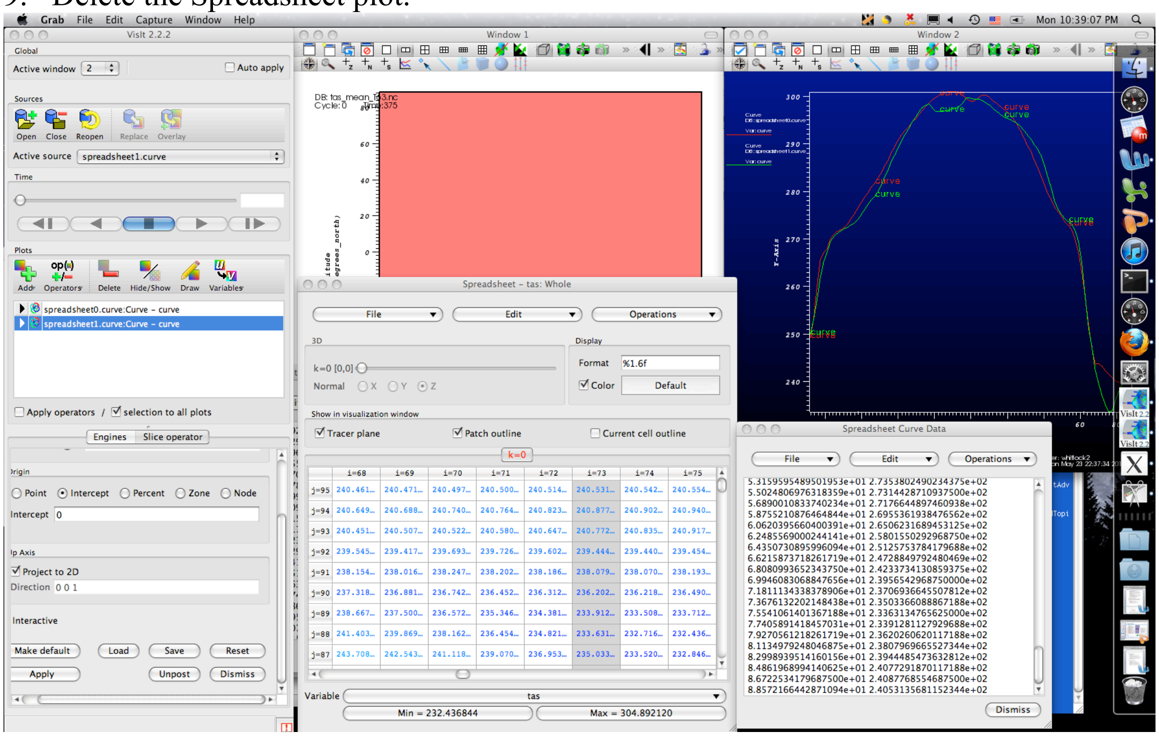
This new operation was achieved by combining different VisIt features via Python scripting.



Exercise group 13: Plots, Selections, and Exporting

Exercise 13a) Spreadsheet plot

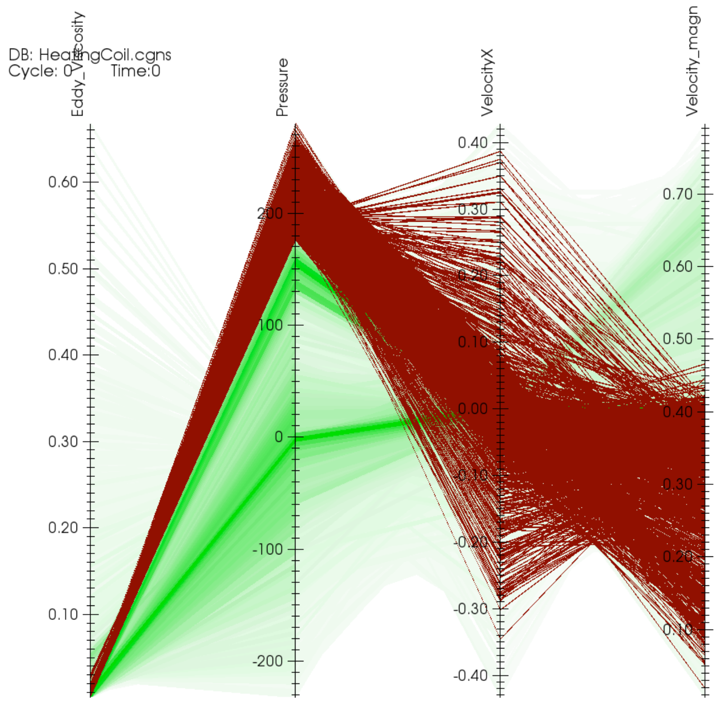
1. Open *tas_mean_T63.nc*.
2. Create a Spreadsheet plot of *tas* and click **Draw**. This will create a new window for the Spreadsheet plot. Note that this new window is the actual plot and not the plot attributes window. If you are plotting structured meshes then the visualization window will show a visual representation of the slice of 2D data that is being displayed by the Spreadsheet plot.
3. Select a range of cells in the **Spreadsheet plot** window. This will enable the **Operations** menu.
4. Select the **Sum** option in the Spreadsheet window's **Operations** menu.
5. Click on a column header to select an entire column.
6. Select **Create curve: column vs. coordinate 1** to extract the column and pair it with Y values from the mesh coordinates. This will bring up a new window that shows the XY pairs.
7. In the new **Spreadsheet Curve Data** window, select **Plot curve** from the window's **Operations** menu. This will plot a curve in a new visualization window.
8. Delete the new window with the Curve plots.
9. Delete the Spreadsheet plot.



Exercise 13b) Parallel Coordinates Plot

1. Open *HeatingCoil.cgns*.
2. Create a Parallel Coordinates plot of *Eddy_Viscosity*.

3. When the Parallel Coordinates wizard opens add *Pressure*, *VelocityX*, and *Velocity_magnitude* to the list of variables that the plot will use.
4. Click **Draw**.
5. Enable the Axis Restriction Tool in the visualization window.
6. Experiment with setting the limits for each axis using the Axis Restriction Tool. You can also set the axis limits in the **Parallel Coordinate plot attributes** window. The plot image below does not depict the location of the axis restriction tool's hot points but they are adjusted to be roughly next to the location of the red focus. Adjust the axis restriction tool until you have limited the focus so it resembles the plot image below.
7. Leave your plots in the visualization window since they will be used in the next exercise.

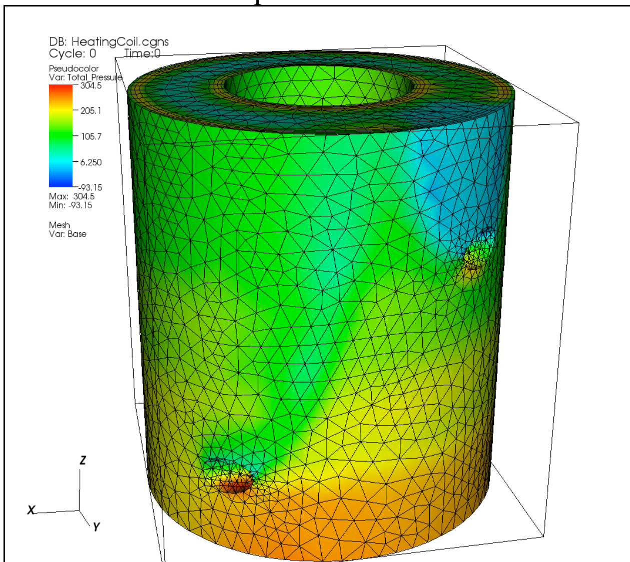
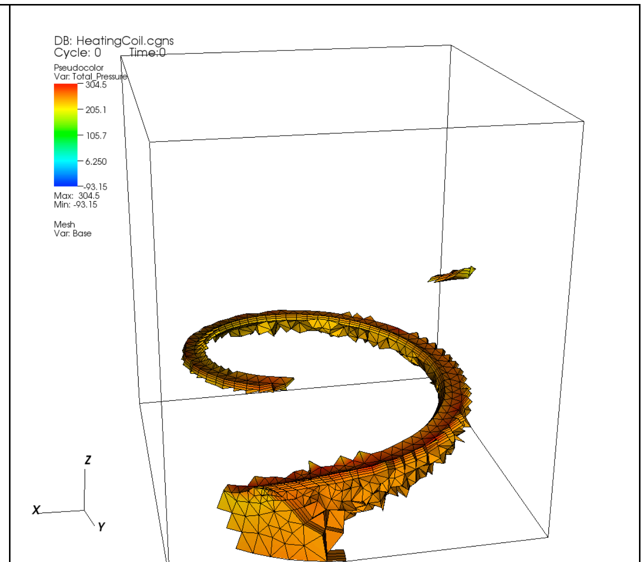


Exercise 13c) Data Selections

Data selections let you save the list of cells used by a plot as a mask that can be applied to other plots to reduce the set of cells that are processed when those secondary plots are generated.

1. Use the Parallel Coordinates plot that was created in the previous exercise.
2. Open the **Selections** window and create a new selection.
3. Turn on the **Automatically apply updated selections** check box.
4. Create a new window and create a Pseudocolor plot of *Total_Pressure* and click **Draw**.

5. Open the **Subset** window and select *selection1* for the **Applied selection**. This will make the selection created by the Parallel Coordinates plot be applied to the Pseudocolor plot.
6. Manipulate the Parallel Coordinates plot using the Axis Restriction Tool and observe the effects on the plot in window 2.
7. Delete window 2
8. Delete the plots in window 1.

	
Original plots	Plots using selection created by the Parallel Coordinates plot

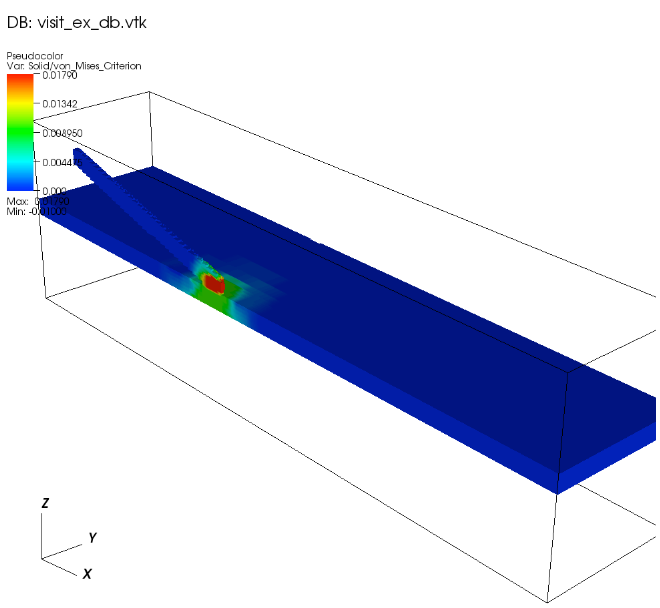
Exercise 13d) Exporting Data (optional)

VisIt can save the data from a plot to a new database using Export database. Exporting a database saves all of the data that was used to generate the plot. This includes internal data that is normally removed by the face list filter before rendering.
You might want to use export database if you plan to:

- Save results of a complex calculation so you can reuse the results without having to recalculate
- Analyze some VisIt data in another application

1. Open *VISITCLASSDATA\velodyne\pelh.*.vld database*.
2. Add a Pseudocolor plot of *Solid/von_Mises_Criterion*.
3. Add a Resample operator and open the **Resample operator attributes** window.
4. Turn off the **Resample Entire Extents** check box.
5. Set **Start X** to *0*.
6. Set **End X** to *23*.
7. Set **Samples in X** to *100*.
8. Set **Start Y** to *0*.
9. Set **End Y** to *4.8*.

10. Set **Start Z** to *-3.8*.
11. Set **End Z** to *2.73*.
12. Set **Samples in Z** to *100*.
13. Set **Value for uncovered regions** *-0.01*.
14. Click **Draw**. You should see a large brick instead of the bullet/plate impact bodies. They are embedded within the brick.
15. Open the **Export** window and select **VTK** for the **Export to** format.
16. Click the **"…"** button and navigate to the *VISITCLASSDATA* directory.
17. Click the **Export** button to save the resampled data as a VTK file.
18. Delete all plots.
19. Open the new *visit_ex_db.vtk* data file.
20. Create a Pseudocolor plot of *Solid/von_Mises_Criterion*.
21. Set the **Limits** to **Use Current Plot**.
22. Apply a Threshold operator since we will use it to remove the "uncovered" values that were inserted into the volume.
23. Set the Threshold **Lower bound** to *0*.
24. Set the **Show zone if** setting to **All in range**.
25. Click **Draw**. The resulting plot should look much like the original dataset except the mesh structure is different.



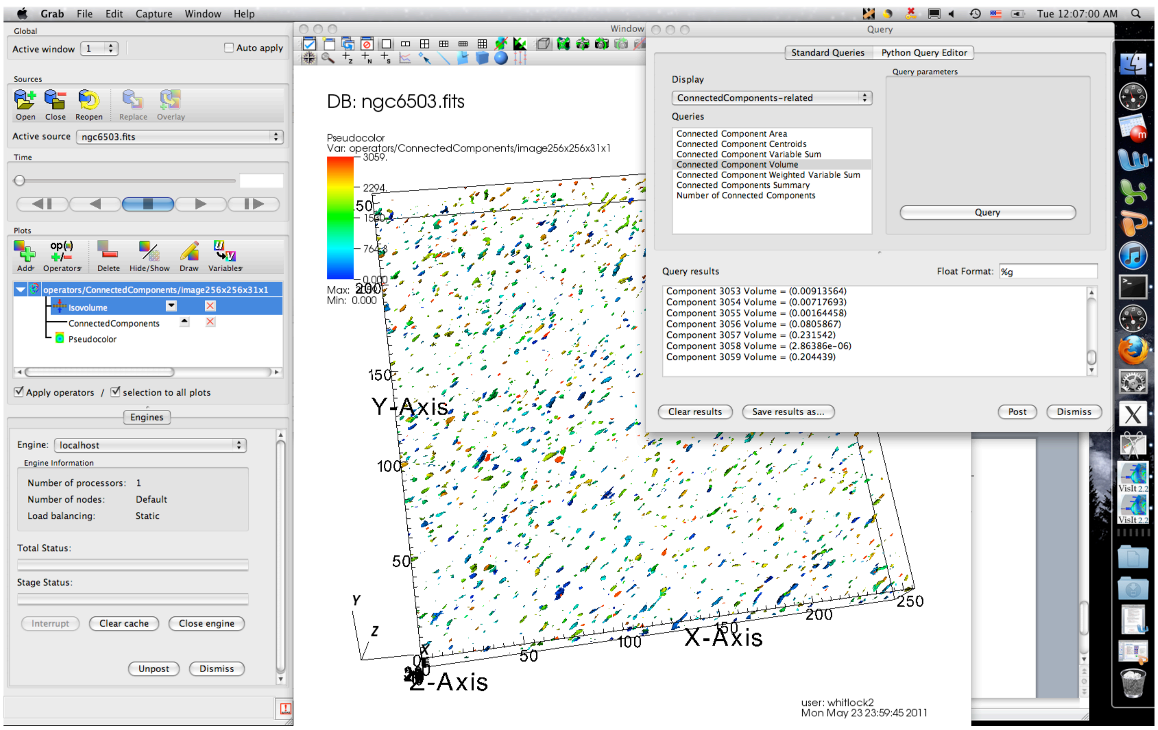
Exercise group 14: More Data Analysis

Exercise 14a) Connected Components

Connected components lets you pick out groups of connected cells.

1. Open *ngc6503.fits.*

2. Create a Pseudocolor plot of *operators/ConnectedComponents/image256x256x31x1*.
3. Add an Isovolume operator and move its position in the operator list to first place.
4. Open the Isovolume operator attributes window and set the **Maximum** value to *-0.005*.
5. Set the Isovolume's variable to *NGC6503*.
6. Click **Draw**.
7. You will see a lot of fragments of various sizes.
8. Open the **Query** window and select the **ConnectedComponents-related** queries.
9. Click on **Connected Component Volume** and run the query by clicking the **Query** button. VisIt will output the volumes of each of the connected components
10. Delete the plot

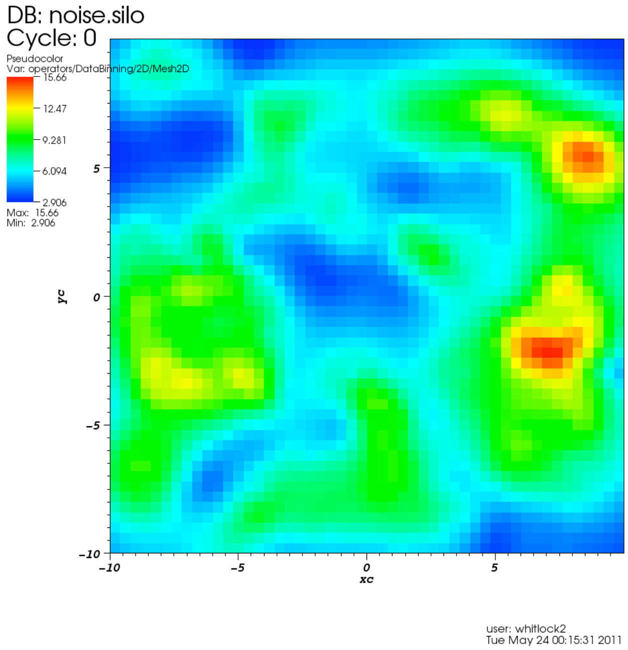


Exercise 14b) Data Binning Operator

Let's sum a 3D dataset's variable along the Z axis.

1. Open *noise.silo*.
2. Create a Pseudocolor plot of *operators/DataBinning/2D/Mesh2D*.
3. Open the **Expressions** window.
4. Create *xc = coord(Mesh)[0]*
5. Create *yc = coord(Mesh)[1]*
6. Open the **DataBinning operator attributes** window
7. Select **2D** for **Dimensions**.
8. On **Dimension 1**, choose *xc* for the variable.

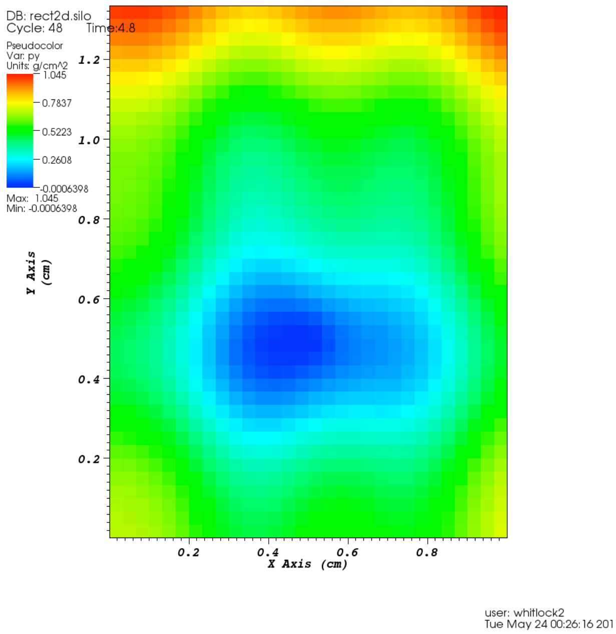
9. On **Dimension 2**, choose *yc* for the variable.
10. For **Reduction Operator**, choose **Sum**.
11. Let's sum the *grad_magnitude* variable so choose that for **Variable**.
12. Click **Draw**.



Exercise 14c) Python Filters

Python filters allow you to create complex expressions using the Python programming language. Python filters enjoy direct access to the VTK datasets that are processed in VisIt as plots are executed.

1. Open *rect2d.silo*.
2. Open the **Expressions** window.
3. Create a new expression *d_wave*.
4. Click on the **Python Expression Editor** tab.
5. Add *d* to the arguments text field.
6. Open *VISITCLASSDATA\filter.py* and paste its contents into the Python expression definition.
7. Create a Pseudocolor plot of *d_wave*.


DB: rect2d.silo
Cycle: 48 Time:4.8
Pseudocolor
Var: py
Units: g/cm^2
1.045
0.7837
0.5223
0.2608
-0.0006398
Max: 1.045
Min: -0.0006398
Y Axis (cm)
X Axis (cm)
user: whitlock2
Tue May 24 00:26:16 2011